December 1992

Report No. STAN-CS-92-1458

PB96-150776

On the Complexity of Partitioning an Assembly

by

R.H. Wilson, J.-C. Latombe, T. Lozano-Perez

Department of Computer Science

Stanford University

Stanford, California 94305

On the Complexity of Partitioning an Assembly

Randall H. Wilson Jean-Claude Latombe
Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

Tomás Lozano-Pérez
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

Abstract

We consider the following problem that arises in assembly planning: given an assembly, identify a subassembly that can be removed as a rigid object without disturbing the rest of the assembly. This is called the assembly partitioning problem. Polynomial-time solutions have been presented when the motions allowed for the separation are of certain restricted types. We show that for assemblies of polyhedra, the partitioning problem for arbitrary sequences of translations is NP-complete. The reduction is from 3-SAT. The proof applies equally when each part in the assembly is limited to a constant number of vertices; when rotations are allowed; when both subassemblies are required to be connected; and for assemblies in the plane where each part may consist of a number of unconnected polygons.

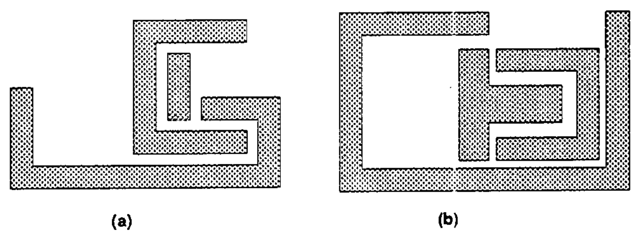

Figure 1: Examples of assemblies requiring non-straight-line motions for disassembly. (a) is a monotone binary assembly while (b) is not.

Introduction

This paper addresses a geometric problem in assembly planning. The problem is: given an assembly of parts, identify a subassembly that can be removed from the assembly. That is, identify a subset of the parts that can be moved (as a single rigid object) to infinity without disturbing the other parts. This is the *assembly partitioning problem.*

The partitioning problem arises in assembly planning. An *assembly sequence* is a sequence of motions that constructs an assembly from its constituent parts, and for rigid parts is the reverse of a *disassembly sequence.* In this paper we assume that the assembly sequences are *binary* and *monotone*, i.e. only one group of parts moves at a time, and the motion completely separates the moved parts from the rest of the assembly. In other words, no parts are placed in intermediate positions. For example, the assembly in figure 1(a) can be assembled by a monotone binary assembly sequence, while the assembly in figure 1(b) cannot. A monotone binary assembly sequence can be found, if one exists for the assembly, by repeated application of an assembly partitioning algorithm.

The vast majority of assemblies in industry are monotone binary, and most assembly planning systems make this assumption as well (see for instance [3]). As a result, the partitioning problem is of great practical importance. Previous work has presented polynomial-time partitioning algorithms for several useful cases when the separating motions are of certain restricted types, such as single translations [1, 11].

In this paper we show that when arbitrary translations are allowed to separate the two subassemblies, the partitioning problem for polyhedral as-

semblies is NP-complete. The proof applies equally when each part in the assembly is limited to a constant number of vertices and when rotations are allowed, as well as to assemblies in the plane where each part may consist of many unconnected polygons. The complexity of the partitioning problem for assemblies of simple polygons remains open.

1 Related Work

Much of the work on geometric separation problems is relevant to assembly planning; for an overview see Toussaint [10]. Pollack, Sharir, and Sifrony [9] give an efficient algorithm to separate two simple polygons in the plane.

In non-monotone assembly sequences, parts may assume many intermediate positions during disassembly. Natarajan [7] and Wolter [13] showed that assembly sequencing is PSPACE-hard when non-monotone sequences are allowed.

In monotone, non-binary assembly sequences, more than one subassembly may move independently at the same time. Palmer [8] considered the infinitesimal, non-binary partitioning problem: determining whether a feasible set of simultaneous, infinitesimal motions exists for the parts of an assembly. He showed that this problem is NP-complete by a reduction from 3-SAT.

Polynomial-time solutions to the binary partitioning problem exist for certain types of restricted disassembly motions. Arkin, Connelly, and Mitchell [1] give a polynomial-time algorithm for partitioning an assembly of polygons in the plane with a single infinite translation, and they have extended the algorithm to polyhedral assemblies in 3D [6]. Wilson [11, 12] solves the 3D partitioning problem in polynomial time when the disassembly motions are restricted to either (a) infinitesimal translations or rotations or (b) infinite translations. In the infinitesimal case. a subassembly is identified that can move an infinitesimal distance, which is only a necessary condition on a removal path.

Lozano-Pérez and Wilson [5] extend Wilson's framework to arbitrary disassembly paths. While their algorithm might prove useful in practice, they do not show a polynomial time bound. In this paper we show that the problem they address is NP-complete. However, when a disassembly path is limited to a constant number of translations, their construction allows a polynomial-time algorithm for partitioning.

2 Complexity of Partitioning

The translational partitioning problem consists of identifying a subassembly of a given assembly that can be separated (as a rigid object) from the rest of the parts by a sequence of translations. In this section we show that translational partitioning for polyhedral assemblies is NP-complete. Section 2.1 shows that translational partitioning is in NP, while sections 2.2 and 2.3 show that it is NP-hard.

Problem 1 (Translational Partitioning) *Given a set A of polyhedra in space, identify a proper subset S of A such that S can be separated from $A \setminus S$ by a collision-free sequence of translations of S.*

2.1 Translational Partitioning is in NP

A first approach to a partitioning problem is to generate all possible ways to divide the assembly A into two subassemblies, then call a path planner to decide if any pair of subassemblies can be separated. For any fixed dimension d, path planning can be performed in polynomial time [4]. A nondeterministic machine can guess the subassembly to remove, then check for a removal path in polynomial time. Therefore translational partitioning is in NP.

2.2 Translational Partitioning is NP-hard

We show that translational partitioning is NP-hard by a reduction from 3-SAT [2]. The proof consists of a set of instructions that show how to construct an assembly of polyhedra that can be partitioned if and only if a given formula in 3CNF has a satisfying truth assignment. A 3CNF formula is a conjunction of clauses, where each clause is a disjunction of 3 literals and each literal is either u_i or $\overline{u_i}$ (i.e. $\neg u_i$) for some variable u_i. For instance, the following formula is in 3CNF form:

$$(\overline{u_1} \vee u_2 \vee u_3) \wedge (u_2 \vee \overline{u_2} \vee \overline{u_3}) \wedge (u_1 \vee u_2 \vee \overline{u_3}) \tag{1}$$

We construct an assembly consisting of $2v + 2$ parts, where v is the number of variables in the 3CNF formula. The assembly resembles a lock with a key inserted, along with $2v$ additional parts representing the truth assignments to the variables in the formula. To partition the assembly, the key must be removed, along with exactly half of the truth assignment parts:

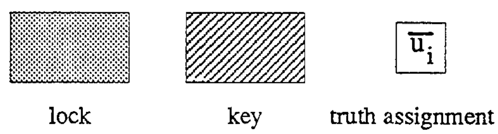

Figure 2: Patterns used for parts in figures

one representing either *true* or *false* for each variable. As they are removed, the key and truth assignments must pass through a series of gates that enforce the clauses of the 3CNF formula. Thus partitioning the assembly is equivalent to finding a truth assignment that satisfies all the clauses.

The proof shows how to construct:

1. A representation for a truth assignment.
2. An assignment construct to require that *true* or *false* is assigned to each variable.
3. A TEST gate to enforce a value of a variable.
4. An OR gate to combine TEST gates into the disjunctive clauses in the formula.
5. An AND mechanism for the conjunction of the clauses.
6. The full assembly combining these elements, that can be partitioned if and only if the formula has a satisfying truth assignment.

The assembly will first be constructed in the plane, where each part may consist of any number of unconnected simple polygons constrained to move rigidly. This assembly will be transformed in section 2.3 into an equivalent assembly of (normal) polyhedra. In the figures that follow, the parts will be drawn as shown in figure 2: polygons belonging to the lock are in gray, polygons belonging to the key are hashed, and the $2v$ truth assignment (TA) parts are all drawn white with labels.

Representing Truth Assignments Truth assignments are represented as shown in figure 3. Part of the key is used as a reference point. For each variable u_i in the 3CNF formula, the truth assignment to u_i is represented by the presence of one of the TA parts u_i or $\overline{u_i}$. Each TA part is 2 units

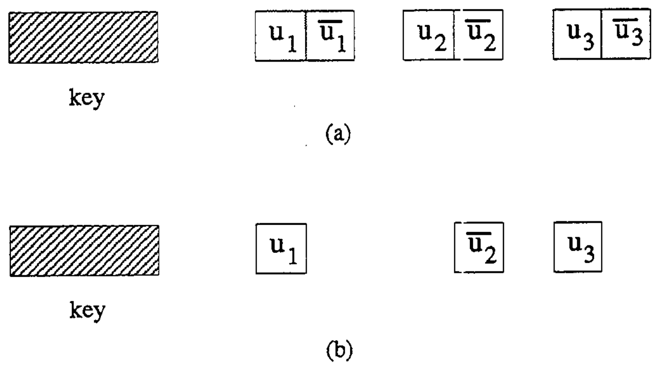

Figure 3: (a) The truth assignment construct and (b) the truth assignment $(u_1, \overline{u_2}, u_3)$

wide, while the key reference is 6 units wide. Figure 3b represents the truth assignment $(u_1, \overline{u_2}, u_3)$.

Assignment Construct The assignment construct is shown in figure 4. It ensures that at least one TA part for each variable must be removed with the key. One 2-unit square (called an *assignment peg*) is added to the key for each variable; remember that the polygons of the key must move together. A guide (part of the lock) forces the key to be removed downward. As the key reference is translated out of the guide, it can shift one unit left or right; this allows the assignment peg for each variable to bypass either u_i or $\overline{u_i}$ but not both. 2 units of vertical space is left between the assignment pegs to allow independent horizontal motion when selecting the truth assignment for each variable. This guarantees that each variable is given a truth assignment, and every truth assignment is possible.

Note that the assignment construct allows both u_i and $\overline{u_i}$ to move with the key. If the resulting subassembly can be removed from the lock, then either truth assignment for u_i satisfies the formula.

A TEST Gate A TEST gate is shown in figure 5. The key and TA parts can pass through the gate if and only if one of the u_i has a specified value. The holes for the other $u_j, j \neq i$ allow both u_j and $\overline{u_j}$ to pass through. Since the key reference is 6 units wide, it cannot pass through any holes but the

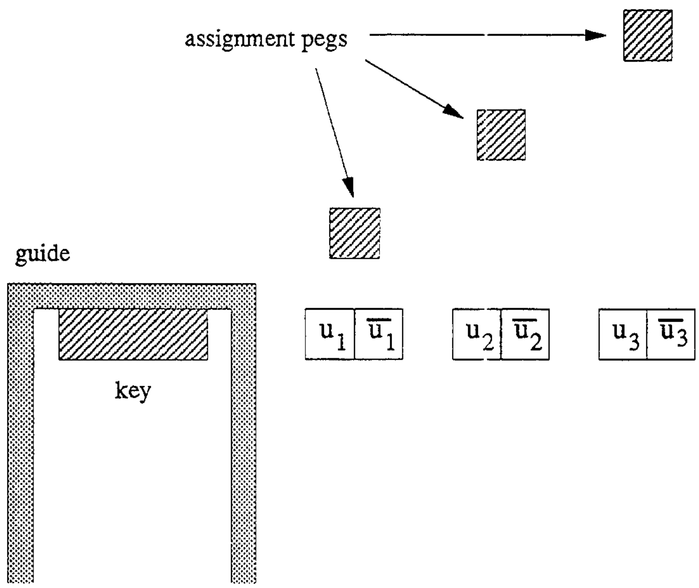

Figure 4: The assignment construct

left one. After the key reference passes through the gate, horizontal motion allows the assignment pegs to pass through also. The TEST gate shown enforces the condition $\overline{u_2}$.

An OR Gate The OR gate for a clause consists of a horizontal wall in the lock, with several TEST gates in the wall (figure 6). The key and TA parts can pass through the OR gate if and only if they can pass through at least one of the TEST gates. The OR gate in figure 6 represents the clause $(\overline{u_1} \vee u_2 \vee u_3)$.

The AND Mechanism The conjunction of all the clauses is represented by placing the OR gates one after another vertically, so that the key must pass through each one in turn to be removed. Enough vertical space must be left to allow the assignment pegs to move fully through the OR gate for one clause before encountering the next.

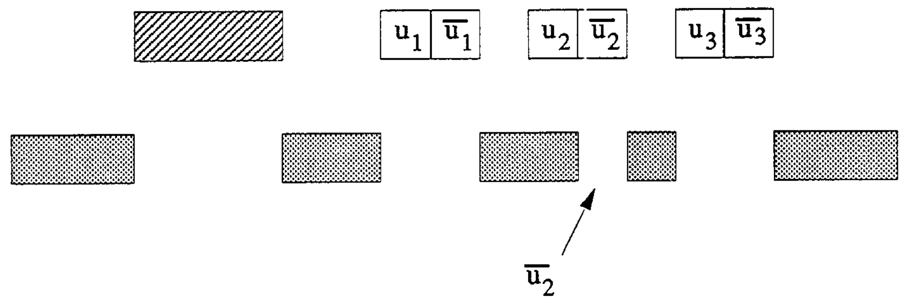

Figure 5: A TEST gate

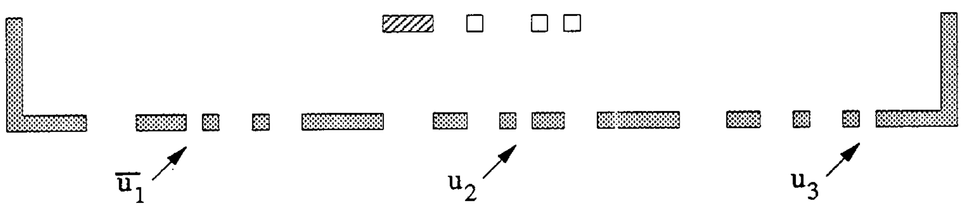

Figure 6: An OR gate

The Assembly Figure 7 shows the whole assembly constructed for the 3CNF formula in equation (1) (8 parts consisting of 46 polygons in this case). A sealing piece is added to the key, ensuring that no parts may be removed without the key. One unit of clearance on each side allows horizontal motion during the assignment phase of removal; after that, the sealing piece is free of the lock, so it does not restrict horizontal motions of the key.

We claim that this assembly can be partitioned if and only if the 3CNF formula has a satisfying truth assignment. Assume that there exists a satisfying truth assignment τ for the formula. Then let S be a subassembly composed of the key and the TA parts representing τ. To satisfy the formula, τ must satisfy at least one of the literals in each clause; for each such literal, the corresponding TEST gate can be passed by S. Therefore S can be moved through each OR gate in turn, and removed.

On the other hand, assume that the assembly can be partitioned. No

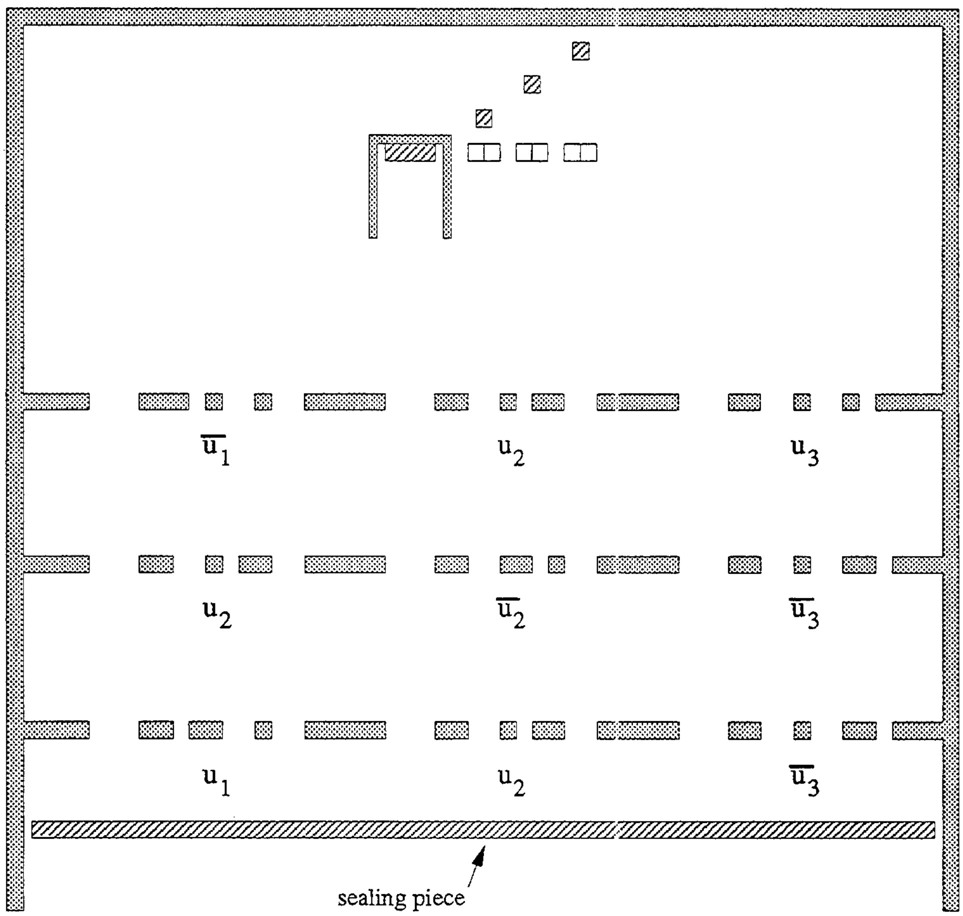

Figure 7: The full assembly

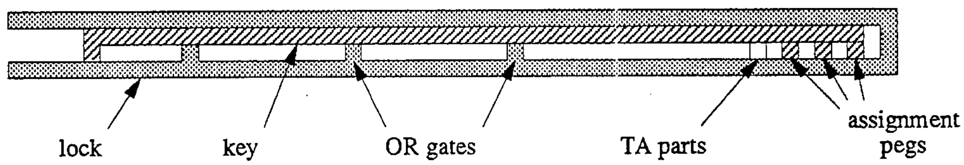

Figure 8: A side view of the polyhedral assembly

parts can be removed without removing the key from the lock. For the key to be removed, one TA part for each variable must be removed as well, since the guide forces the assignment pegs to collide with u_i or $\overline{u_i}$ for each i. The resulting subassembly must then be moved through the OR gates representing each of the clauses in turn. If such a subassembly can be removed, then the TA parts in it give a satisfying truth assignment for the formula. Therefore the assembly can be partitioned if and only if the 3CNF formula has a satisfying truth assignment.

2.3 Polyhedra

An assembly of (normal) polyhedra can be constructed in the same way as above. Note that only two parts (the lock and the key) in the 2D assembly have unconnected components; these parts are connected in the polyhedral assembly. The polyhedral assembly consists of four layers:

1. The bottom layer is a single plate added to the lock, covering the bottom of the assembly and connecting all pieces of the lock.

2. The next layer is the same as the 2D assembly of figure 7.

3. The third layer is a plate connecting all the pieces of the key, but not extending horizontally any more than the key reference and assignment pegs (thus allowing horizontal freedom).

4. The top layer is the same as the bottom layer, enclosing the lock from above.

Figure 8 shows a side view of the polyhedral assembly. It can be partitioned only by removing the key, and the only interesting motions are planar ones.

Let v be the number of variables and f be the length of the 3CNF formula in the 3-SAT instance. Each TEST gate can be constructed with

$O(v)$ vertices, and there are $O(f)$ such gates in the assembly. Hence the assembly in figure 8 has complexity $O(vf)$. It can clearly be constructed in polynomial time in v and f. Since an instance of 3-SAT can be reduced to an instance of translational partitioning in polynomial time, translational partitioning is NP-hard. We now have the following theorem.

Theorem 1 *Translational partitioning is NP-complete.*

In addition, the above assembly can be assembled if and only if it can be partitioned. An NP machine could guess all $n-1$ partitionings to disassemble an n-part assembly, then check each in polynomial time, so the problem is clearly in NP.

Corollary 2 *Monotone binary assembly sequencing for polyhedra is NP-complete.*

3 Variants

In the above construction, two of the parts have unbounded complexity, i.e. a part can have a number of vertices that is polynomial in the size of the 3CNF formula. In this section we show that the result holds when the parts are limited to constant complexity. In addition, we describe several variants of the problem to which the construction can be adapted, and identify an important open problem for planar assemblies.

3.1 Parts of Constant Complexity

An equivalent assembly can be constructed such that each part has constant complexity. The basic structures of the key and lock are built of blocks such as those shown in figure 9a; each block connects to adjacent ones with pins. The key reference and assignment pegs are made of these blocks. A surrounding piece holds the key blocks together until the assembly is partitioned, at which time they can be removed vertically and then disassembled (figure 9b). Similarly, the guide and gates of the lock are built of blocks (figure 10a), which are placed inside a box and cover (figure 10b). The key passes through a hole in the cover. The cover traps the lock blocks until the key is removed, after which the cover and lock blocks can be removed and disassembled.

The resulting assembly can be partitioned exactly when the previous assembly can be. A polynomial number of parts are required to construct

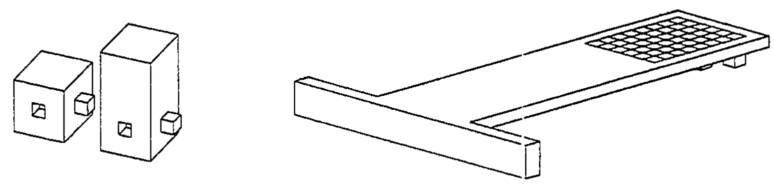
(a) (b)

Figure 9: (a) the blocks, and (b) the key constructed with the blocks

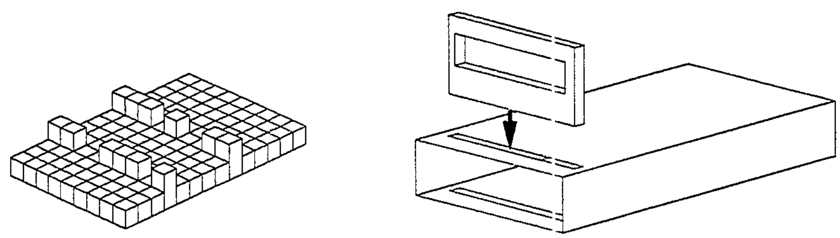
(a) (b)

Figure 10: (a) the lock structure constructed of blocks, and (b) the box and cover enclosing the lock

it. The parts with the most vertices are the blocks (figure 9a), which can be built with 32 vertices each.

Theorem 3 *Translational partitioning for assemblies of polyhedra with at most k vertices (for $k \geq 32$) is NP-complete.*

3.2 Rotations

The construction can be modified to remove all interesting rotations. The sealing piece disallows almost all rotation during the assignment phase of the removal. The key reference is changed to be 8 units tall, so that it must pass through all gates in a horizontal position. In addition, the section of wall connecting TEST gates is made long enough that turning the key and

TA parts 180 degrees does not allow passage. The resulting assembly can be partitioned by translation and rotation exactly when it can be partitioned with translations.

3.3 Connected Subassemblies

For polyhedral assemblies A, another version of the partitioning problem requires that both the removable subassembly S and $A \setminus S$ be connected, i.e. that the union of each set of parts form a connected set. In practice, this is useful constraint for assembly planning. The key and the lock are both in contact with all of the TA parts in the polyhedral assembly above, so the construction applies to the connected partitioning problem.

3.4 Planar Assemblies

In showing the complexity of polyhedral partitioning above, we constructed an assembly for a strange planar case. That is, if a single part may consist of a number of disconnected polygons in the plane, then assembly partitioning in the plane is NP-complete. While this result may prove to be significant in its own right, it raises the following question: what is the complexity of partitioning for assemblies of simple polygons in the plane? This seems to be the most interesting open problem to pursue in the future.

Acknowledgments

The authors thank Dan Halperin and Lydia Kavraki for comments on an earlier draft. The first author was supported by a grant from the Stanford Integrated Manufacturing Association. The second author was partially supported by a grant from DARPA under ONR contract N00014-92-J-1809. The third author was partially supported by a grant from DARPA under ONR contract N00014-91-J-4038.

References

[1] E. M. Arkin, R. Connelly, and J. S. B. Mitchell. On monotone paths among obstacles, with applications to planning assemblies. In *Proc. of the 5th ACM Symp. on Computational Geometry*, pages 334–343, 1989.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, New York, 1979.

[3] L. S. Homem de Mello and S. Lee, editors. *Computer-Aided Mechanical Assembly Planning.* Kluwer Academic Publishers, Boston, 1991.

[4] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[5] T. Lozano-Pérez and R. H. Wilson. Assembly sequencing for arbitrary motions. Manuscript submitted to 1993 IEEE Intl. Conf. on Robotics and Automation, September 1992.

[6] J. S. B. Mitchell. Personal communication, December 1990.

[7] B. K. Natarajan. On planning assemblies. In *Proc. of the 4th ACM Symp. on Computational Geometry*, pages 299–308, 1988.

[8] R. S. Palmer. *Computational Complexity of Motion and Stability of Polygons.* PhD thesis, Cornell Univ., 1989.

[9] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete and Computational Geometry*, 3:123–136, 1988.

[10] G. T. Toussaint. Movable separability of sets. In G. T. Toussaint, editor, *Computational Geometry.* Elsevier, North Holland, 1985.

[11] R. H. Wilson. *On Geometric Assembly Planning.* PhD thesis, Stanford Univ., March 1992. Stanford Technical Report STAN-CS-92-1416.

[12] R. H. Wilson and J.-C. Latombe. On the qualitative structure of a mechanical assembly. In *Proc. of the National Conf. on Artificial Intelligence*, pages 697–702, 1992.

[13] J. D. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly.* PhD thesis, The Univ. of Michigan, 1988.